高度なRAGシステム (iRAG)

概要

このシステムは、文書検索とSQL検索を統合した高度なRAG（Retrieval-Augmented Generation）システムです。Streamlitベースの直感的なUIを提供し、Azure OpenAI Serviceを活用して、多言語対応の自然言語処理による強力な情報検索と質問応答を実現します。データ（CSV/Excel）に対するSQL検索も同時にします。

主な機能

- **ハイブリッド検索**: ベクトル検索とキーワード検索を組み合わせ、Reciprocal Rank Fusion (RRF) によって検索精度を向上させます。
- **日本語特化**: SudachiPyによる日本語形態素解析を使用し、日本語の文書処理に最適化されたハイブリッド検索を実現します。
- **Text-to-SQL**: 自然言語のクエリを解析し、アップロードされたCSV/Excelファイルからのデータベースファイルに対して自動的にSQLクエリを生成します。
- **専門用語辞書 (Golden-Retriever)**: アップロードから専門用語をその定義を抽出し、辞書を構築。この辞書を用いてクエリの理解を深め、よりコンテキストに沿った回答を生成します。
- **評価システム**: Recall、Precision、MRR、nDCG、Hit Rateなどの指標でRAGシステムの検索精度を定量的に評価できます。
- **ユーザーフレンドリーなUI**: タブ構成のインターフェース、メッセージ履歴、ドキュメント管理など、使いやすいストリームリットアプリケーションと洗練された設計です。

システム構成

システムは大きく以下のコンポーネントから構成されています：

```
.
├── app.py                      # Streamlitアプリケーションのエントリポイント
├── requirements.txt            # 必要なPythonライブラリ
├── .env.example                # 環境変数の設定テンプレート
├── rag/                        # RAGシステムのコアモジュール
│   ├── __init__.py
│   ├── chains.py               # LangChainのチェーンとプロンプト設定
│   ├── config.py               # 設定ファイル(Config)
│   ├── evaluator.py            # 評価システムモジュール
│   ├── ingestion.py            # ドキュメントの取り込みと処理
│   ├── jargon.py               # 専門用語辞書の管理
│   ├── retriever.py            # ハイブリッド検索リトリーバー
│   ├── sql_handler.py          # Text-to-SQL機能の処理
│   └── text_processor.py       # 日本語テキスト処理
├── rag_system_enhanced.py      # RAGシステムのファサード
├── evaluate_rag.py             # 評価スクリプト
├── scripts/                    # 拡張したスクリプト
│   ├── term_extract.py
│   └── term_extractor_embeding.py
├── state.py                    # Streamlitのセッション状態管理
├── ui/                         # UIコンポーネント
```

```
│   ├── __init__.py
│   ├── chat_tab.py
│   ├── data_tab.py
│   ├── dictionary_tab.py
│   ├── documents_tab.py
│   ├── settings_tab.py
│   └── sidebar.py
└── utils/                      # ユーティリティ関数・モジュール
    ├── __init__.py
    ├── helpers.py
    └── style.py
```

インストール手順

1. **仮想環境の作成と有効化**:

```
python -m venv myenv
source myenv/bin/activate   # Linux/macOS
myenv\Scripts\activate       # Windows
```

2. **依存関係のインストール**:

```
pip install -r requirements.txt
```

3. **環境変数の設定**: `.env.example` ファイルをコピーして `.env` ファイルを作成し、以下の設定を記入してください。最低限、以下の設定が必要です。
   - `AZURE_OPENAI_API_KEY`
   - `AZURE_OPENAI_ENDPOINT`
   - `AZURE_OPENAI_CHAT_DEPLOYMENT_NAME`
   - `AZURE_OPENAI_EMBEDDING_DEPLOYMENT_NAME`
   - `PG_URL` (PostgreSQLの接続URL) または `DB_*` の各項目

使い方

以下のコマンドでStreamlitアプリケーションを起動します。

```
streamlit run app.py
```

評価システムの使用方法

RAGシステムの検索精度を評価するには、以下のスクリプトを実行します：

```
python evaluate_rag.py
```

評価機能の特徴

- **複数の評価指標**:
  - Recall@K: 関連文書の再現率
  - Precision@K: 検索結果の精度
  - MRR (Mean Reciprocal Rank): 平均逆順位
  - nDCG (Normalized Discounted Cumulative Gain): 正規化減損累積利得
  - Hit Rate@K: ヒット率
- **複数の類似度計算手法**:
  - Azure Embedding: 埋め込みベースの類似度
  - Azure LLM: LLMベースの類似度判定
  - Text Overlap: テキストの重複度
  - Hybrid: 複数手法の組み合わせ
- **柔軟な評価方法**:
  - CSVファイルからの評価データ読み込み
  - プログラムでの直接評価
  - 結果のCSVエクスポート

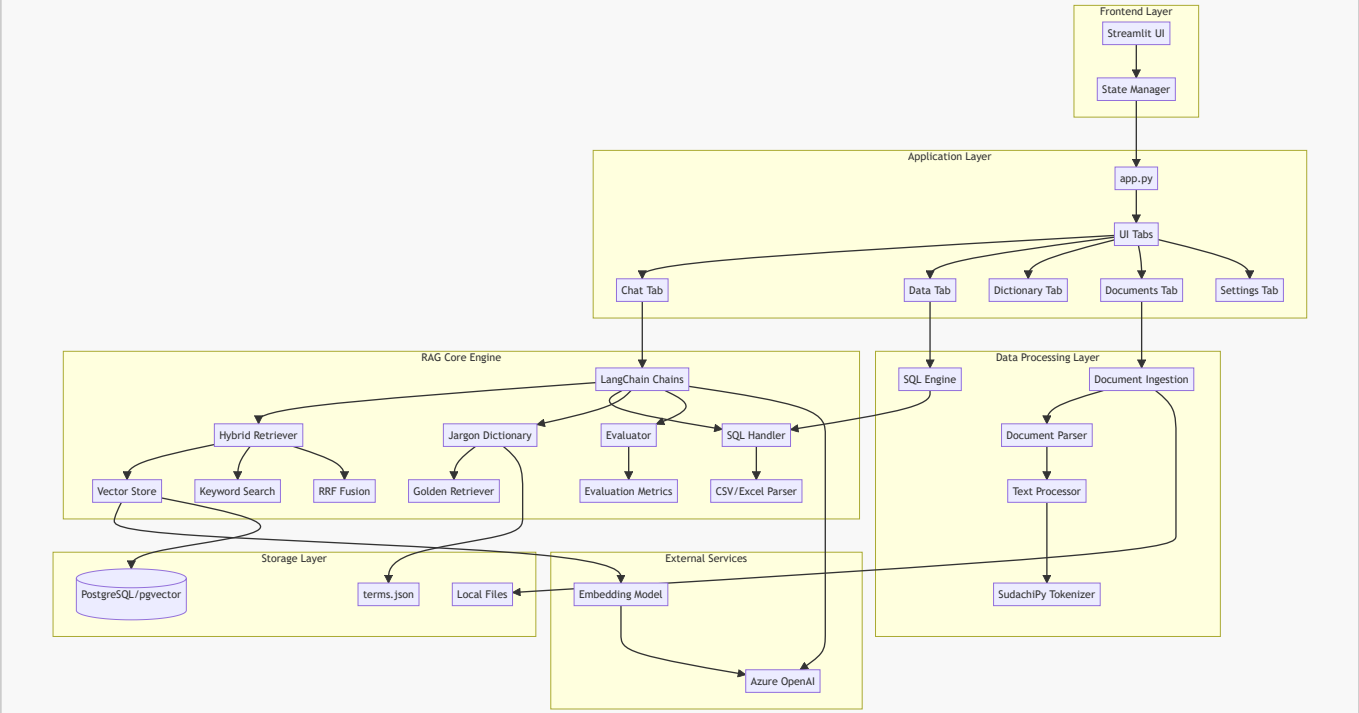
評価データの形式

CSVファイルは以下の形式で準備してください：

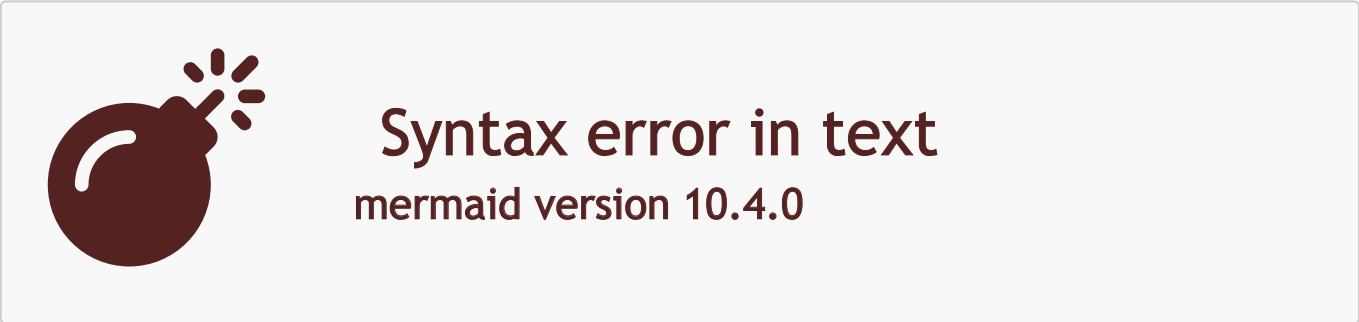
- 質問: 評価用の質問
- 想定の引用元1, 想定の引用元2, ...: 期待される情報源
- チャンク1, チャンク2, ...: 検索結果（オプション）

アーキテクチャ概要

システム全体構成

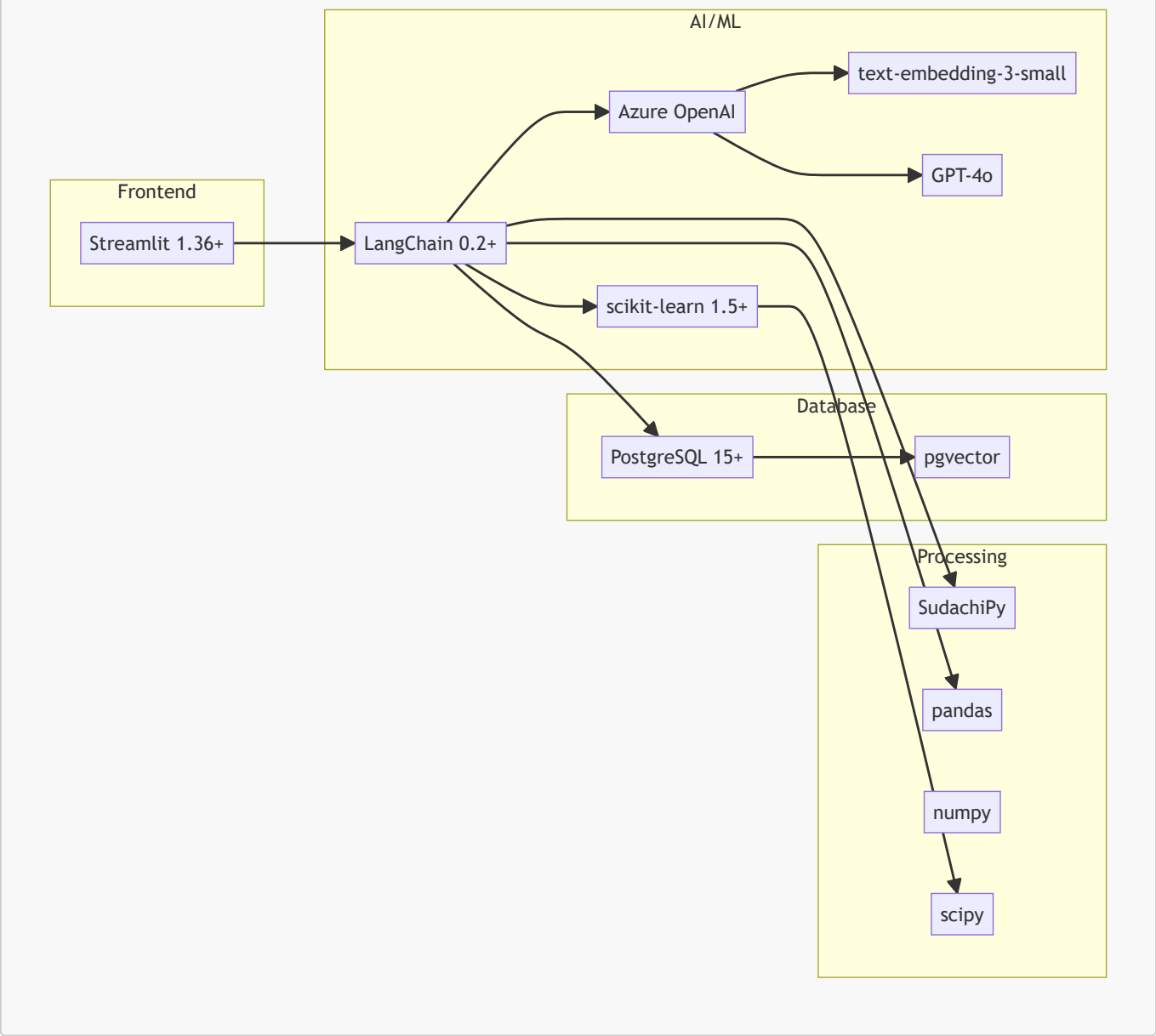


データフロー



クラス構成

主要技術スタック



ライセンス

このプロジェクトはMITライセンスの下で公開されています。

貢献

プルリクエストや問題報告を歓迎します。大きな変更を行う場合は、まずissueを開いて変更内容について議論してください。

サポート

問題が発生した場合や質問がある場合は、GitHubのissuesセクションで報告してください。